A perception and manipulation system for collecting rock samples[1]

T. Choi, H. Delingette, M. DeLuise, Y. Hsin, M. Hebert, K. Ikeuchi

The Robotics Institute,
Carnegie Mellon University,
Pittsburgh Pa15213

Abstract

An important goal of a planetary exploration mission is to collect and analyze surface samples. As part of the CMU Ambler project, we are investigating techniques for collecting samples using a robot arm and a range sensor. The aim of this work is to make the sample collection operation fully autonomous. We describe in this paper the components of the experimental system that we have developed, including a perception module that extracts objects of interest from range images and produces models of their shapes, and a manipulation module that enables the system to pick up the objects identified by the perception module. We have tested the system on a small testbed using natural terrain.

1 Introduction

One of the most important goals of a planetary exploration mission is to collect and analyze terrain samples. As part of the CMU Ambler project [2], we are investigating techniques for autonomously collecting samples. We have developed a system that is able to collect small rocks using computer vision and planning. Our goal is to eventually integrate the system to the Ambler system, a six-legged autonomous robot for planetary exploration.

We have developed a rock sampling system that includes: a robot arm, a range finder, and a small terrain mock-up that contains sand and small rocks. The goal of the rock sampling system is to identify, locate, and pickup rocks from the terrain. The control flow of the rock sampling system is shown in Figure 2: First an range image of the scene is taken and features are extracted from the image (Section 2). The features are surface features such as surface discontinuities that are used to extract the object boundaries. Then the contours of the objects in the scene are extracted. Since, we are dealing with natural environments, we make very weak assumptions on the possible shapes of the objects and on the distribution of the features in the image. To handle those constraints, we have developed a new shape extraction algorithm (Section 3.1) based on the concept of deformable contours. The set of points enclosed by the contour of an object is approximated by a superquadric surface (Section 3.2). In some cases the object representation using superquadrics may not be sufficient. An algorithm based on deformable surfaces can extract directly a surface representation of an object using the image features without relying on superquadric fitting (Section 4). Finally, the parameters of the surface that approximate each object (superquadric or deformable surface) are used to grasp it using a clam-shell gripper (Section 6). The algorithms for object extraction assume that there is an initial guess of the positions of the objects in the image. We present an algorithm for selecting the object location hypothesis automatically in Section 5.

[1]This research was sponsored by NASA under Grant NAGW 1175. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the US Government.

2 Image acquisition and feature extraction

In order to manipulate objects, we need an accurate description of their shape. This implies that we need to use a sensor that can sense the 3-D surfaces observed in a scene. Therefore, the only possibility is to use a sensor that measures range data. Many range sensing techniques are available [3]. The range sensor that we are currently using is an active sensor that consists of a projector equipped with a computer-controlled LCD screen and a camera [14]. The projector illuminates the scene through the LCD screen. As several illuminations patterns are projected, the corresponding images of the scene are collected by the camera. The range to each point in the scene is recovered from the shape of the projected patterns. The output of the sensor is set of four 256×256 images: an intensity image and three images, X, Y, and Z that contain the three coordinates of the three spatial coordinates of each pixel. The coordinates are with respect to a fixed reference frame defined at calibration time. The spatial and range resolution of this sensor is appropriate for this application in which we need high-resolution measurements at close range. We currently use the intensity image for only display purposes although it could also be used in the object extraction algorithms [12].

Figures 3 and 4 show the images of two scenes. The upper left image is the intensity image, the other three images are the coordinate images. The coordinate images are coded on 16 bits and displayed on 8 bits which accounts for the periodic effect in those images. Figure 5 shows a 3-D display of the data from Figure 3.

Once an image is acquired, the next step is to extract features of the terrain that can help extract the objects of interest in the environment. Many different types of features can be extracted from range data [4] ranging from planar facets to local extrema of the principal curvatures [5]. However, most of those techniques do not apply to this problem mainly because we are working in an unconstrained natural environment which rules out all the feature types, *e.g.* planar or quadratic patches, that assume a known geometric structure of the environment. Furthermore, it is our experience that the standard techniques based on curvature analysis perform well only when the data is very accurate and well distributed. We have chosen an approach in which we detect local features that are relatively insensitive to noise. We do not force the features to provide a complete description of the terrain, in particular we do not expect those features to connect to each other to form the boundaries of the objects in the scene. Instead, we want each feature to give partial evidence of the presence of an object in its vicinity. Grouping the detected features into objects is the job of the segmentation algorithms introduced in the next Section.

Three types of features are extracted:

- Range shadows: Objects produce shadows in the range images, which are areas of the scene that are illuminated by the projector but that are not visible from the camera because they are occluded by an object's surface. This phenomenon occurs with any sensor that uses a triangulation technique. Range shadows are therefore a important cue for the extraction of objects. Extracting the range shadows does not require any image processing since they are identified by the sensor itself.
- Surface discontinuities: A surface discontinuity is a large variation of range between neighboring points in the image. Such discontinuities occur mostly in the vicinity of the occluding edge of an object. Surface discontinuities are detected by applying an edge detector to the image of the range values, $r = \sqrt{x^2 + y^2 + z^2}$. The final edges are obtained by thresholding the resulting edge magnitude. The threshold is computed from the distribution of the edge magnitudes in a large window centered at each image pixel. The reason for using a variable threshold is that the range r varies more rapidly as points are measured further from the sensor. Spurious edges would be detected if a fixed threshold were used.
- Surface normal discontinuities: Surface normal discontinuities occur when two surfaces intersect as is the case when an object is resting on top of the terrain. The normal discontinuities are detected by first computing the unit surface normal n at each point and by finding the low values of the dot products $n_1 \cdot n_2$ of the surface normals at adjacent pixels. The three coordinate images must be smoothed first since the surface normal computation is quite sensitive to noise in the data. Further smoothing is applied to the surface normals.

The image pixels that are labeled as one of the three feature types are grouped into connected regions. The set of feature regions is the input to the segmentation algorithms. Figure 6 shows the features computed from the image of Figure 3. The features are shown as shaded regions. As expected, the features are concentrated around the objects although some are detected on the underlying terrain and no group of features form a closed object boundary.

3 Object extraction: deformable contours and superquadrics

The features give an indication of where the boundaries of the objects may be located in the scene. However, the raw features are not sufficient for reliably extracting the objects from the scene because the objects may be small or partially buried in the terrain. Therefore, we cannot use a simple region extraction that would assume that the features are grouped into closed boundaries. Instead, we used the concept of deformable contours and deformable surfaces. The idea is that a contour that is attracted by 2-D forces generated by the detected features and by the data points measured on the terrain is iteratively deformed until the forces applied to it are in equilibrium. A smoothness constraint is added to the forces so that the contour or the surface does not have sharp discontinuities of orientation or curvature. The final product is a smooth contour that approximates the shape of an object that is partially enclosed by features. The advantage of this approach is that object descriptions can be extracted from the image even if only few scattered features are observed. This is in sharp contrast with other vision problems such as model-based object recognition in which an accurate model of the objects is known apriori. We do not make any assumption on the shape of the objects other than a maximum and minimum object size, and we do not make any assumption on the configuration of the features.

This approach is inspired from Witkin's "snakes" [11] and from Terzopoulos' symmetry-seeking surfaces [16]. We describe in detail the deformable contours algorithm in the next Section.The algorithm assumes that one point that lies inside the object is initially selected. The actual selection of this starting point is the object of Section 5. We assume for now that this point is available. Once a contour is extracted, a three-dimensional model of the corresponding set of points must be built. We use superquadrics to represent the object models (Section 3.2).

3.1 Deformable contours

A deformable contour is a contour in a range image that is subject to forces that change its shape over time. The contour reaches a stable shape when all the forces are in equilibrium. The points that are inside the region enclosed by the final contour are used to described the shape of the object. The algorithm used to derive a shape representation from the region is described in Section 3.2.

We represent a contour by an ordered set of pixel (r_i, c_i) where r_i is the row coordinate in the image, and c_i is the column coordinate. A 3-vector p_i, that is the position of the scene point measured at pixel (r_i, c_i), is associated with each pixel. In addition, the normal to the contour n_i is defined at each p_i. The n_i's are two-dimensional vectors expressed in image coordinates. Furthermore, n_i is always oriented from the inside to the outside of the contour. It is always possible to define such an orientation since the contour is guaranteed to be closed without self-intersections. Each p_i is subject to a set of forces. Each force is a signed scalar that indicates in which direction p_i is attracted. A positive force indicates that p_i is attracted toward the outside of the contour in the direction of the nearest feature. The algorithm is designed in such a way that the contour can only grow outward.

Each pixel of the contour is subject to two types of forces 7(a). The external forces are exerted by entities that are not part of the contour such as features. The internal forces depend on the contour itself and are independent of the data. Internal forces are typically used to force the contour to be as smooth as possible.

The first external force is generated by the features. It is an attractive force defined at each point p by:

$$F_{\text{feature}} = \sigma_{\text{feature}}\left(\frac{\|p - \mathcal{F}(p)\|}{R_{max}}\right) \quad (1)$$

where $\mathcal{F}(p)$ is the point of the image features that is the closest to p, σ_{feature} is a function that relates the force to the distance between contour point and feature (Figure 7(b)), and R_{max} is the maximum expected object size. The closest point $\mathcal{F}(p)$ is calculated by searching the feature points along 16 directions around the contour normal. Since this is a potentially expensive operation, we use several constraints to limit the search: First, the features that are too far from the contour point are not considered. Second, we use the fact that the order in which features appear around an object is defined by the geometry of the sensor and can be computed beforehand thus eliminating features that cannot be part of the current object.

The second type of external force is generated by the starting point. Its purpose is to prevent the contour from "overgrowing" by generating an attractive force towards the center point. The force is defined by:

$$F_{\text{center}} = \sigma_{\text{center}}\left(\frac{\|p - p_0\|}{R_{max}}\right) \quad (2)$$

where R_{max} is defined as before, p_0 is the starting point, and σ_{center} is the attraction function (Figure 7(c)).

The purpose of the internal force is to guarantee that the contour is reasonably smooth. The idea is to make the shape of the contour close to an ellipse. To do that we approximate the contour by an ellipse $\mathcal{E}$ of equation $(p - p_{\mathcal{E}})^t A(p - p_{\mathcal{E}}) = 1$, where $p_{\mathcal{E}}$ is the center of the ellipse, and A is a 2×2 symmetrical matrix. The distance between p and $\mathcal{E}$ is defined by:

$$D(p, \mathcal{E}) = \frac{|(p - p_{\mathcal{E}})^t A(p - p_{\mathcal{E}}) - 1|}{2\|A(p - p_{\mathcal{E}})\|} \tag{3}$$

$D(p, \mathcal{E})$ is an approximation of the Euclidian distance between p and $\mathcal{E}$. The internal force is defined by:

$$F_{\text{internal}} = \sigma_{\text{internal}}\left(\frac{D(p, \mathcal{E})}{K}\right) \tag{4}$$

where σ_{internal} is the attraction function (Figure 7(d)), and K is a constant that controls how far from an ellipse the contour is allowed to be. In practice $K = 0.4$.

The contour deforms itself iteratively. At each iteration, the internal and external forces are computed at each point. Each point is moved according to the resulting force, The complete algorithm follows two steps:

1. Initialize: The initial contour is a small contour centered at the starting point.
2. Iterate: The following steps are iterated until the contour does not deform itself significantly.
   - At each point p_i compute the sum of the forces: $F = F_{\text{feature}} + F_{\text{center}} + F_{\text{internal}}$.
   - p_i is moved by one pixel in the direction of the nearest feature point $\mathcal{F}(p_i)$ if $F > 0$.
   - Resample the contour after all the contour points have been moved according to the forces.
   - Estimate the best-fit ellipse $\mathcal{E}$.

Provided that there is a reasonable starting point, this algorithm produces object contours that are quite good approximations of the true object contour even if the features are very sparse. Figure 8 shows the regions that have been found for each object in the scene using the feature of Figure 6. The starting points were selected automatically using the algorithm of Section 5.

3.2 Superquadrics

Once regions corresponding to objects have been segmented out using the deformable contour algorithm, the corresponding set of 3-D points must be grouped into a surface representation. The resulting object models are used to compute grasp position and manipulator motion.

Although one could use the set of 3-D points computed by the segmentation directly, we use superquadrics to represent the objects. Superquadrics are generalizations of quadric surfaces [1] that can represent a wide variety of shape. Using superquadrics present several advantages: First, it is a compact representation that allows us to represent a wide range of surfaces using a small set of parameters. Second, it provides a global representation of an object whose surface is only partially visible. Lastly, the parameters of a superquadric surface are easily recovered from the coordinates of a set of points.

Superquadrics are described by an implicit equation $F(x, y, z) = 1$, where:

$$F(x, y, z) = \left(\left(\left(\frac{X}{a_1}\right)^{\frac{2}{\varepsilon_2}} + \left(\frac{Y}{a_2}\right)^{\frac{2}{\varepsilon_2}}\right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{Z}{a_3}\right)^{\frac{2}{\varepsilon_1}}\right)^{\varepsilon_1} \tag{5}$$

and where (X, Y, Z) are the coordinates of (x, y, z) after transformation by a rigid transformation that defines the position and orientation of the superquadric, and a_1, a_2, and a_3 are the sizes of the superquadric along the three directions. Superquadrics can represent a variety of shapes form cubes to ellipsoids by varying the two "roundness" parameters ε_1 and ε_2. Other parameters such as bending and tapering can be included in the equation. To recover the superquadric from a set of points, we use the Levenberg-Marquart minimization approach suggested by Solina [1]. In this approach, the input set of points is first approximated by an ellipsoid which constitutes the starting point of the minimization, then an error function of the form:

$$E = \sum_{(x,y,z) \text{ data point}} (F(x, y, z) - 1)^2 \tag{6}$$

is minimized with respect to the parameters of the superquadric. This approach works well in our case in which a dense set of points is measured on a portion of the surface (see [13] or [9]) for other superquadric fitting techniques).

Figure 9 shows the superquadric models of the objects found in Figure 8. The models are displayed as wireframes superimposed on the intensity image.

4 Object extraction: deformable surfaces

Deformable contours extract the objects by using essentially the geometry of the scene in the image plane. The result is a region in the image that has to be processed further to yield a complete description of the object. A more direct, although more costly, approach would be to directly find the closed surface that best approximates the data, that is the 3-D points measured on the terrain and the detected features. This leads to the idea of deformable surfaces which are smooth closed surfaces that are subject to forces from the terrain and the features. As with the deformable contours, the surface deforms itself until it closely fits the observed shape. The advantage is that the resulting closed surface should provide all the information needed to pick up the object. As in the case of deformable contours, the algorithm assumes that an initial point is selected inside each object.

The algorithm operates on discrete data, images and discrete features. However, for the sake of clarity it is best to think first of the case of a *continuous* deformable surface that is subject to forces and deforms itself over time. It can be shown that such a surface would reach a stable equilibrium when the Lagrangian of the system of forces reaches a minimum according to the principle of least action [8]. A similar application of the principle can be found in [17]. The Lagrangian is defined by: $L = T - U$ where T is the integral of the kinetic energy over time and U is the integral of the potential energy. If the surface is parametrized as $x = x(\eta, \omega, t)$, $y = y(\eta, \omega, t)$, $z = z(\eta, \omega, t)$, where t is the time, and (η, ω) are the parameters of the surface, then the problem is to find the function that minimizes L. This is a variational problem that can be solved by applying Euler's equation. To simplify the notations, we will denote the points of the surface by $r(\eta, \omega, t)$, r being the 3-vector (x, y, z), and we will denote the partial derivatives by using subscripts (*e.g.* $r_\omega = \frac{\partial r}{\partial \omega}$). Furthermore, we assume that the parameters η and ω vary between 0 and 1.

The term T depend only on the kinetic energy and can be written as:

$$T = \int_0^{t_0} \int_0^1 \int_0^1 \mu \|r_t\|^2 d\omega d\eta dt \tag{7}$$

where μ is a weighting factor that characterizes the inertia of the surface.

In our case, the surface should be deformed so that the following constraints are satisfied: The surface should be smooth, the surface should be as close as possible to the surrounding features, and the surface should be close to the points measured on the terrain. To satisfy those constraints, the potential energy term U is decomposed into three components:

$$U = U_{\text{smoothness}} + U_{\text{features}} + U_{\text{terrain}} \tag{8}$$

The term $U_{\text{smoothness}}$ encapsulates the constraint that the surface should be smooth and continuous. Formally it is defined by:

$$U_{\text{smoothness}} = \int_0^{t_0}\int_0^1\int_0^1 \alpha_1\left(\|r_\omega\|^2 + \|r_\eta\|^2\right) + \alpha_2\left(\|r_{\omega\omega}\|^2 + \|r_{\eta\eta}\|^2 + 2\|r_{\omega\eta}\|^2\right) d\omega d\eta dt \tag{9}$$

The weights α_1 and α_2 control how much importance is given to the smoothness constraint. The surface can have any arbitrary shape if they are equal to zero, on the other hand contributions from the features and the terrain are ignored if they are very large.

The term U_{features} implements the constraint that the surface should be as close as possible to the surrounding features. In order to define it, we first have to define the distance between a point and a feature. In order to do that, we represent each feature $\mathcal{F}$ by the 3-D polygonal approximation of its skeleton which is a set of 3-D line segments. The distance between a point r on the deformable surface and a feature $\mathcal{F}$ is the distance between r and its projection on the set on line segments that describes $\mathcal{F}$. We denote the projection by $r(\mathcal{F})$. Strictly speaking, we should compute the distance between r and all the points of $\mathcal{F}$. Since this is too demanding computationally, we use the polygonal approximation which allows us to compute the projection directly. With this definition of $r(\mathcal{F})$, we define:

$$U_{\text{features}} = K\int_0^{t_0}\int_0^1\int_0^1 \sum_{\mathcal{F}} S(r,t,\mathcal{F})\|r - r(\mathcal{F})\|^2 d\omega d\eta dt \tag{10}$$

where the sum is taken over all the features $\mathcal{F}$, and where K is a weighting factor. If we think of a set of springs linking each point of the surface to each feature, K would be the stiffness of the springs. The attraction exerted by the features is basically proportional to the squared distance between the point and the feature $\|r - r(\mathcal{F})\|^2$. An attenuation factor $S(r,t,\mathcal{F})$ is added to avoid one undesirable effect of the pure spring model: points that are very far from the features are always subject to very strong forces. What we would like instead is to have a strong attraction to all the points to initiate the deformation and to have the strength of the attraction decrease over time. For a given distance $\|r - r(\mathcal{F})\|$, this is equivalent to vary the stiffness of the spring as a function of time. Furthermore, it is undesirable for the features to apply an arbitrary large force to the points that are far away. We need a cutoff distance over which points are not attracted. We define the correction factor by:

$$S(r,t,\mathcal{F}) = \sigma\left(1 - \frac{t_0}{t_0 - t}\frac{\|r - r(\mathcal{F})\|^2}{r_0^2}\right) \tag{11}$$

where the function σ varies from 0 at $-\infty$ to 1 at $+\infty$ (Figure 10). σ implements the idea of a cutoff distance: points that are too far away from the feature are not attracted. The cutoff distance is given by the normalizing term r_0. In addition to the cutoff distance, for a given distance $\|r - r(\mathcal{F})\|$, the term $\frac{t_0}{t_0 - t}$ makes the stiffness of the spring vary over time: The spring is strong a $t = 0$ and weakens over time until it eventually disappears at $t = t_0$ at which time only the smoothness constraint and the attraction from the terrain are taken into account.

Another way to look at equation is to consider the term $K\|r - r(\mathcal{F})\|^2$ as a fitting term in that it forces the surface to be as close as possible to the features, and to consider the term $S(r,t,\mathcal{F})$ as a segmentation term in that it takes into account only the group of features that is close to the starting point. The last term of the potential U_{terrain} reflects the attraction between the surface and the terrain. It is defined as:

$$U_{\text{terrain}} = \beta\int_0^{t_0}\int_0^1\int_0^1 \frac{1}{\|r - r(T)\|} d\omega d\eta dt \tag{12}$$

where $r(T)$ is the data point that is closest to the surface point r. Since the term inside the integral would become arbitrarily large as the surface moves closer to the data, we introduce a cutoff distance D at which the potential stops increasing. The potential is therefore redefined as:

$$\begin{cases} \frac{1}{\|r - r(T)\|} & \text{if } \|r - r(T)\| \geq D \\ \frac{1}{D} & \text{if } \|r - r(T)\| \leq D \end{cases} \tag{13}$$

This potential implements a gravity force that increases as points move closer. This has the effect that the feature term is dominant initially when the surface is far from the observed terrain while the terrain becomes dominant as the surface moves closer to the terrain. Notice that strictly speaking we should take into account the contributions from *all* the data points in the computation of the force applied to a single surface point. Since this is computationally untractable we limit ourselves to the closest data point.

We now have a definition of the function L given a set of features and a set of points measured on the terrain. The problem is now to find the surface $r(\eta,\omega,t)$ that minimizes L. The solution is found by straightforward application of Euler's equation. We obtain the differential equation:

$$\mu r_{tt} = \frac{1}{2}\beta P(r) + \alpha_1(r_{\omega\omega} + r_{\eta\eta}) - \alpha_2(r_{\omega\omega\omega\omega} + 2r_{\omega\omega\eta\eta} + r_{\eta\eta\eta\eta}) + KS(r,t,\mathcal{F})(r - r(\mathcal{F})) + KS_1(r,t,\mathcal{F})\|r - r(\mathcal{F})\|^2 \tag{14}$$

where S_1 is computed from the first derivative of σ: $S_1(r,t,\mathcal{F}) = -\sigma'\left(1 - \frac{t_0}{t_0 - t}\frac{\|r - r(\mathcal{F})\|^2}{r_0^2}\right)\frac{t_0}{t_0 - t}\frac{r - r(\mathcal{F})}{r_0^2}$, and P is the gradient of the potential due to the terrain attraction, that is the integrand of U_{terrain}: $P(r) = \frac{r - r(T)}{\|r - r(T)\|^3}$.

Applying Euler's equation solves the problem in the case of a *continuous* surface subject to the attraction of the features and the terrain and to a smoothness constraint. To actually compute a solution to the resulting differential equation, we need to construct a discrete approximation of both the surface, that is a discretization of the parameter space (η,ω) and of the time t. Let us consider first the case of the parameter space. Using a straightforward discretization of η and ω in regular intervals of $[0,1]$ would lead to serious problems at the edges of the parameter space just like sampling a sphere along the meridians and parallels leads to problems at the two poles. Since it is not desirable to be forced to handle special cases in the discretization, we would like to use a representation of the parameter space that is as uniform as possible. To do that, we first create a unit sphere that is tesselated using the icosahedron decomposition [6, 7], each point M_i of the tesselation is parametrized by its spherical coordinates (η_i,ω_i) and is a sample point on the surface. The tesselation of the sphere has the property that it is very uniform and that it does not exhibit any poles. With this representation the integrals become sums over the sample points, for example the integral with respect to (η,ω) in U_{terrain} becomes:

$$\sum_{M_i \text{ sample point}} \frac{1}{\|r(\eta_i,\omega_i) - r(T)\|} \tag{15}$$

The time axis is also discretized: the deformation of the surface is implemented as an iterative process, the discrete time is simply the iteration number. With those discrete representations of (η, ω) and t, the derivatives involved in the final solution are approximated by the appropriate finite differences. In particular r_{tt} is given by a combination of the values of r at iterations $t, t-1$, and $t+1$: $r_{tt} = r(t+1) - 2r(t) + r(t-1)$. Replacing r_{tt} by its discrete approximation in the differential equation, we can express the surface at iteration $t+1$, that is the vectors $r(\eta_i, \omega_i)$, as a function of the surface at the two previous iterations t and $t-1$. If F is the right-hand side of 14, we have:

$$r(t+1) = r(t) + (r(t) - r(t-1)) + F \tag{16}$$

After initialization, the deformable surface is iteratively updated using this relation.

To summarize, the algorithm can divided into two steps:

1. Initialize:
   - Extract the terrain features: shadows, discontinuities, normal discontinuities. Compute the polygonal approximations of the skeleton of the features.
   - Generate the discretization of the parameter space by computing a uniform sampling (η_i, ω_i) of the unit sphere.
   - Generate an initial surface. The initial surface is a sphere, that is $r_i = C + Ru_i$, where C is the starting point that is inside the object, R is the radius of the smallest object that we expect to extract, and $u_i = (r_i - C)/\|r_i - C\|$. The algorithm for selecting C is described in Section 5.
2. Iterate until the number of iteration is greater than t_0
   - For each point r of the surface, compute the projections $r(\mathcal{F})$ and $r(\mathcal{T})$.
   - Compute the derivatives of r with respect to η and ω using finite differences.
   - Compute the update term F using Equation 14.
   - Update the surface using Equation 16.

The result of the deformable surface algorithm is illustrated in Figure 11: The upper left part of the figure shows the features overlaid in white on top of the intensity image of a small scene. The upper right part shows a 3-D view of the terrain with the polygonal approximations of the features. The bottom three images show the evolution of the shape of the approximating surface as the algorithm proceeds.

5 Automatic object selection

We have assumed so far that a point is chosen inside each object to initiate the object segmentation process both in 2-D and 3-D. This point should be qualitatively "close to" the center of the object. The question of finding those initial points still remains. The simplest solution is to have an operator interactively select a point in the observed image. This would be acceptable in a teleoperated mode with the appropriate user interface. However, it would be more useful to be able to automatically compute the starting points from the input images. Since there is no prior constraint on where the objects may be in the scene, the only information that we can use are the features and a geometric model of the sensor. Specifically, the automatic segmentation is based on the observation that the presence of an object generates a shadow region in the range image. Therefore, the objects in the scene should be "near" the shadow regions extracted from the range image. The meaning of "near", that is the position of an object with respect to its shadow, is given by the sensor model.

The geometry of the problem is shown in Figure 12. For the sake of clarity, this geometry assumes a one-dimensional sensor; the reasoning can be extended without difficulties to a 2-D sensor: A projector P illuminates the scene while a camera C observes the illuminated scene. We assume that a sensor model provides the coordinates of P and C in a common coordinate system. An object in the scene creates a shadow region between points A and B, corresponding to illumination directions L_A and L_B that are known from the measured coordinates of A and B and from the sensor model. Based on this geometry, the occluding object must be within the dashed region R. A starting point for the 2- or 3-D snakes can be computed by taking the center of that region. It is important to note that this algorithm does not give us the center of the (unknown) object but rather a point that is enough inside the object for the object extraction algorithm to work.

The geometry is similar with a 2-D sensor except that the two points A and B are now contours. In practice, two corresponding points A and B are chosen on the shadow contour and the region R is identified using the 1-D geometry. The starting point S is selected within R at some nominal distance D from A. D is chosen based on the average expected radius of the objects in the scene and based on the minimum and maximum sizes of objects that we can handle given a gripper configuration. Those are reasonable criteria since there is no point in segmenting out objects that we cannot manipulate. D is also used to remove small shadow regions, presumably due to noise, and large regions, generated by objects too large to handle.

The key to automatic object extraction is an accurate geometric model of the sensor that allows us to compute the hypothesized position of objects in the scene based on observed shadow regions. We have implemented this technique using a model of our current sensor. However, it is important to note that the algorithm can be generalized to any range sensor provided that a geometric model exists. We are in the process of modifying the algorithm in order to use an existing geometric sensor modeling system [10]. This will lead to a largely sensor-independent segmentation program.

6 Manipulation

Once we have extracted object descriptions, either superquadrics or deformable surfaces, the last step is to grasp the object. Many different types of gripper design and grasping strategies are possible. The choice of a particular type of grasping is dictated by the analysis of the task. Assuming that the objects to be sampled are mostly isolated and are resting on a soft surface, *e.g.* sand, the grasping task has the following characteristics:

- The objects are far enough from each other. No collision occurs between the gripper and the neighboring rocks.
- We can allow the collision between the gripper and the neighboring sand. This is because
  - damaging the neighboring sand grains is not important,
  - the collision between the gripper and neighboring sand does not cause the configuration of the rock to change.
- we do not know the exact shape of a rock beforehand.

Based on the characteristics of the task and the possible grasping strategies [15], we have selected the spherical grasping strategy using a clam-shell gripper. The gripper has two hemispherical jaws that can close around the object. Using a surface representation of the objects, the grasping strategy is as follows: the center of the gripper is first aligned with the center of mass of the surface, then the gripper is rotated so that the jaws are parallel to the main axis of the surface. Finally the gripper is lowered until the jaws are in contact with the terrain surrounding the object. The object is grasped by closing the two jaws. Figure 13 show the gripper and the grasp operation.

This approach works well under the stated conditions. However, we need tighter control of the grasping operation than is provided by

the spherical grasping in more difficult environments (*e.g.* Figure 4). In this case, we will use the object model calculated from the deformable surfaces algorithm conjunction with a three-finger gripper. The object model is more accurate than the superquadric model, and the three-finger gripper allows for more flexibility in the grasping. The price to pay is in longer computation time, and in more complex gripper design and control.

7 Conclusion

We have developed a testbed for sampling in unstructured terrain, that is the identification and manipulation of small natural objects. We have implemented the complete cycle of perception, representation, and manipulation. The objects are extracted from range images from surface features using either deformable contours or deformable surfaces. The objects can be represented by superquadric surfaces and by discrete surfaces. The system has been demonstrated in real natural environment using a manipulator equipped with a clam-shell gripper.

Our current work concentrates on building a more complete description of the terrain by using multiple images, hierarchical representation of the observed scenes, and by using more accurate object description such as deformable surfaces. We are working on a three-finger gripper to perform manipulation in a cluttered environment. Finally, we are exploring strategies for modifying the terrain using the manipulator to facilitate the sampling operations.

The sampling system currently resides on a small testbed. We want to eventually move it to a real vehicle, and to demonstrate the interaction between navigation and sampling, thus providing a complete system for planetary exploration.

References

[1] R. Bajcsy and F. Solina. Three Dimensional-object Representation Revisited. Technical Report MS-CIS-87-19, Univeristy of Pennsylvania, Department of Computer and Information Science, March 1987.

[2] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker. An Autonomous Rover for Exploring Mars. *IEEE Computer*, (6), June 1989.

[3] P.J. Besl. Range Imaging Sensors. Technical Report GMR-6090, Computer Science Department, GM Research Laboratories, Warren, Michigan, March 1988.

[4] P.J. Besl and R.C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, March 1985.

[5] M. Brady, J. Ponce, A. Yuille, and H. Asada. Describing surfaces. In *Proc. 2nd International Symposium on Robotics Research*. MIT Press, Cambridge, MA, 1985.

[6] C. Brown. Fast display of well-tessellated surfaces. *Computer and Graphics*, 4(4):77–85, April 1979.

[7] J.D. Clinton. Advanced structural geometry studies, part I: polyhedral subdivision concepts for structural applications. Technical report, NASA, September 1971.

[8] C. Fox. *An introduction to the calculus of variations*. Dover Publications Inc., 1963.

[9] A. Gupta, L. Bogoni, and R. Bajcsy. Quantitative and Qualitative Measures for the Evaluation of the Superquadric Models. In *Proc. IEEE Workshop on Interpretation of 3-D Scenes*, November 1989.

[10] K. Ikeuchi and J. C. Robert. Modeling Sensor Detectability with VANTAGE Geometric/Sensor Modeler. In *Proc. of DARPA Image Understanding Workship*, pages 721–746. Science Application, Inc., May 1989. (a slightly longer version is available as CMU-CS-89-120).

[11] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *Intern. Journal of Computer Vision*, 2(1):321–331, 1988.

[12] Y.P. Lim. *Shape Recognition in the Rocks World*. PhD thesis, Dept. of Electrical Engineering and Computer Science, MIT, April 1988.

[13] A. Pentland. On the extraction of shape information from shading. Technical Report Vision science technical report 102, MIT Media Lab, March 1988.

[14] K. Sato, H. Yamamoto, and S. Inokuchi. Range imaging system utilizing nematic liquid crystal mask. In *International Conf. on Computer Vision*, pages 657–661, London, 1987.

[15] C.L. Taylor and R.J. Schwarz. The Anatomy and Mechanics of the Human Hand. In *Artificial Limbs*, pages 22–35, 1955.

[16] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-Seeking Models and 3D Object Recognition. *Intern. Journal of Computer Vision*, 1(1):211–221, 1987.

[17] S.W. Zucker, C. David, A. Dobbins, and L. Iverson. The organization of curve detection: coarse tangent fields and fine spline coverings. In *Proc. Second Intl. Conf, on Computer Vision*, Tampa, December 1988.

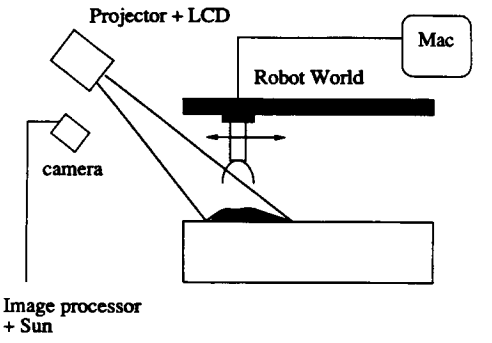


Figure 1: Testbed

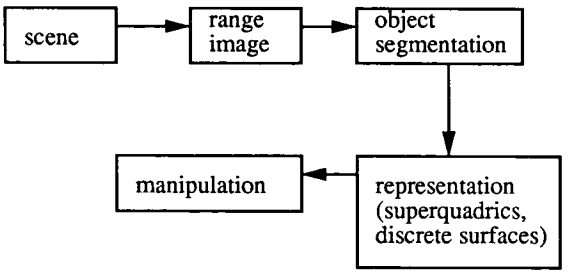


Figure 2: Control flow

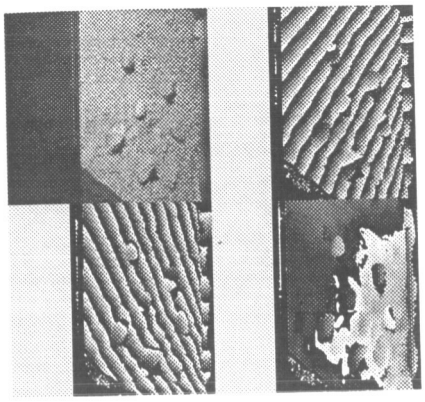


Figure 3: Range image of a scene

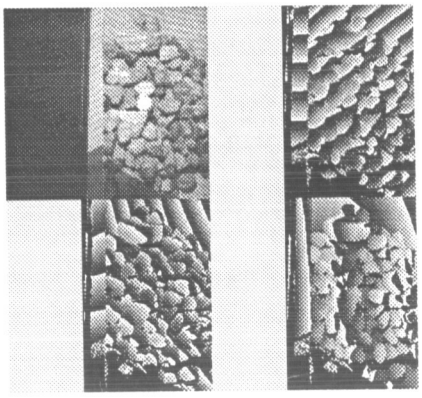


Figure 4: Range image of a complex scene

Figure 5: 3-D view of the data of Figure 3

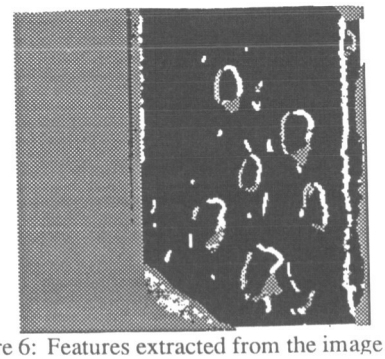
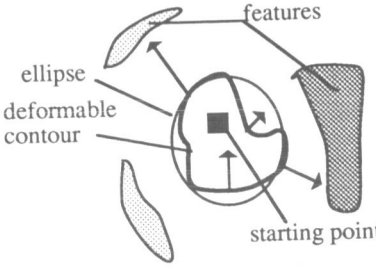
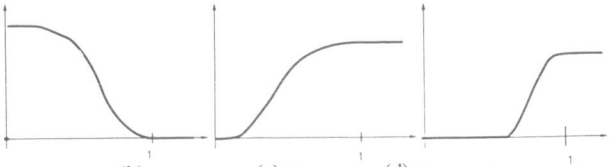


Figure 6: Features extracted from the image of Fig. 3



(a) Forces applied to a deformable contour



(b) $\sigma_{feature}$ (c) σ_{center} (d) $\sigma_{internal}$

Figure 7: 2-D segmentation algorithm

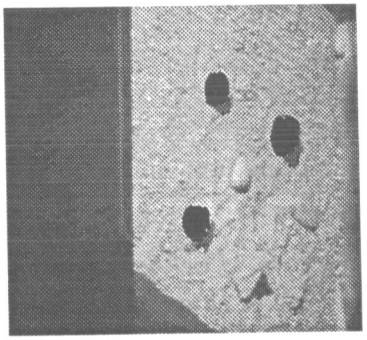


Figure 8: 2-D segmentation algorithm

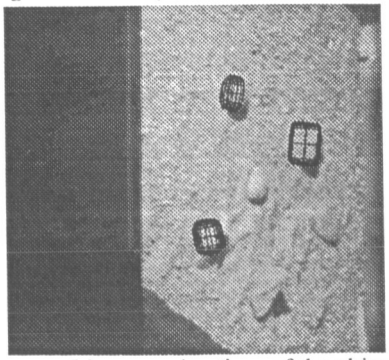


Figure 9: Superquadric approximations of the objects of Fig. 8

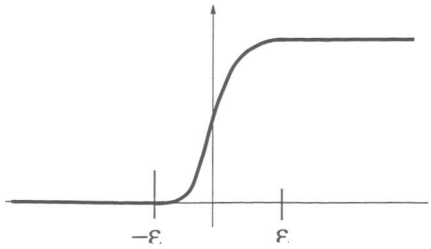
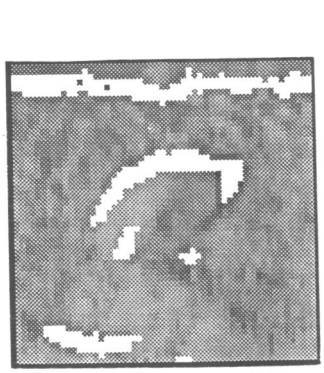
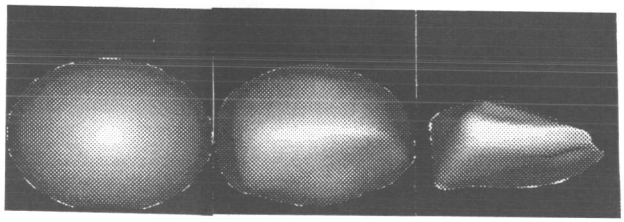


Figure 10: Sigma function



(a) Image of a scene and features

(b) 3-D view of the terrain



(c) Initial shape (d) Intermediate shape (e) Final result

Figure 11: 3-D segmentation algorithm

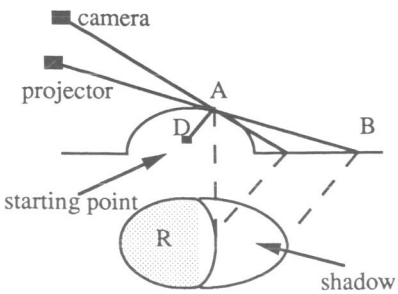
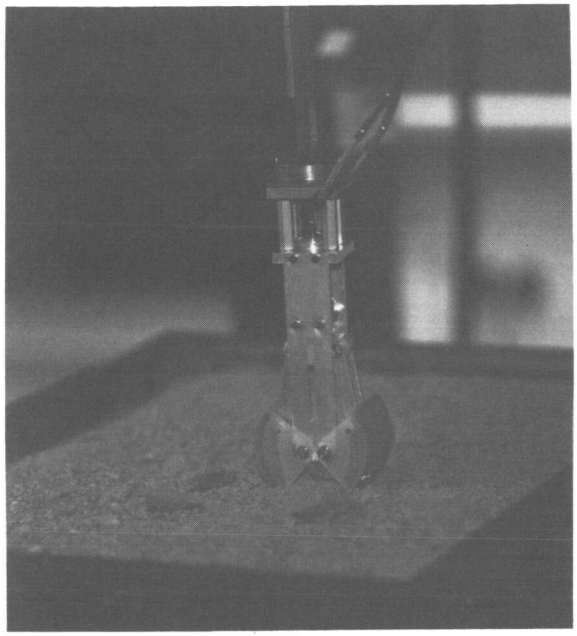


Figure 12: Automatic object selection



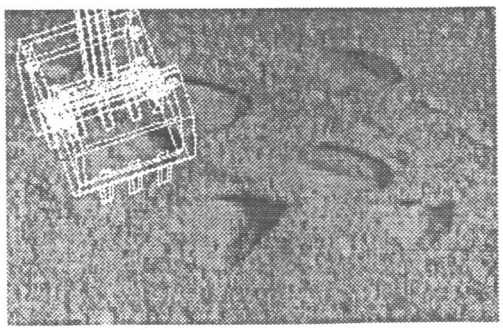


Figure 13: Gripper and grasp strategy